# SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

**An Autonomous Institution**

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

## DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COURSE NAME : 19EC513 – IMAGE PROCESSING AND COMPUTER VISION

III YEAR / V SEMESTER

**Unit V- Computer Vision**

**Topic : Geometric Primitives and Transformation**

## Geometric primitives

Geometric primitives form the basic building blocks used to describe three-dimensional shapes. In this section, we introduce points, lines, and planes

**2D points**. 2D points (pixel coordinates in an image) can be denoted using a pair of values,

$\mathbf{x} = (x, y) \in \mathcal{R}^2$, or alternatively,

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}. \quad (2.1)$$

**2D lines**. 2D lines can also be represented using homogeneous coordinates ˜l = (a, b, c).

The corresponding *line equation* is

$$\bar{\mathbf{x}} \cdot \tilde{\mathbf{l}} = ax + by + c = 0. \quad (2.3)$$

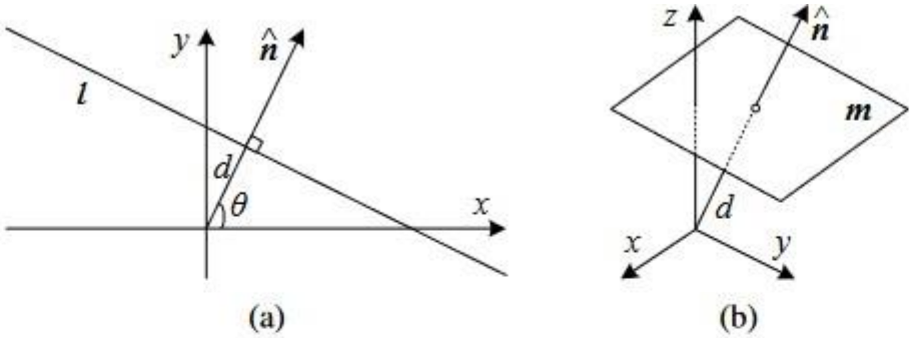



**Figure 2.2** *(a) 2D line equation and (b) 3D plane equation, expressed in terms of the normal $\hat{\mathbf{n}}$ and distance to the origin d.*

**2D conics.** There are other algebraic curves that can be expressed with simple polynomial homogeneous equations. For example, the conic sections (so called because they arise as the intersection of a plane and a 3D cone) can be written using a quadric equation

$$\tilde{\mathbf{x}}^T \mathbf{Q} \tilde{\mathbf{x}} = 0.$$

**3D points**. Point coordinates in three dimensions can be written using inhomogeneous co-ordinates x = (x, y, z) ∈ R3 or homogeneous coordinates ˜x = (˜x, ˜y, ˜z, ˜w) ∈ P3. As before, it is sometimes useful to denote a 3D point using the augmented vector ¯x = (x, y, z, 1) with

$$\tilde{\mathbf{x}} = \tilde{w}\bar{\mathbf{x}}.$$

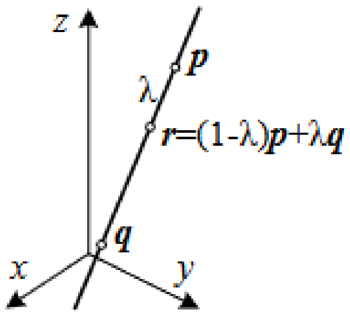



**Figure 2.3** *3D line equation,* $\mathbf{r} = (1 - \lambda)\mathbf{p} + \lambda\mathbf{q}$.

**3D planes.** 3D planes can also be represented as homogeneous coordinates ˜m = (a, b, c, d) with a corresponding plane equation

$$\bar{\mathrm{x}} \cdot \tilde{\mathrm{m}} = ax + by + cz + d = 0.$$

**3D lines**. Lines in 3D are less elegant than either lines in 2D or planes in 3D. One possible representation is to use two points on the line, (p, q). Any other point on the line can be expressed as a linear combination of these two points

$$\mathbf{r} = (1 - \lambda)\mathbf{p} + \lambda\mathbf{q},$$

**3D quadrics**. The 3D analog of a conic section is a quadric surface

$$\bar{\mathrm{x}}^T \mathrm{Q} \bar{\mathrm{x}} = 0$$

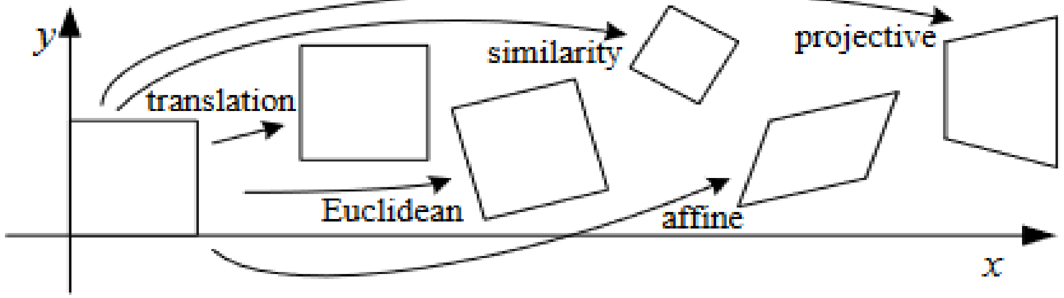



**Figure 2.4** *Basic set of 2D planar transformations.*

**Translation**. 2D translations can be written as

$$x' = \begin{bmatrix} I & t \end{bmatrix} \bar{x}, \qquad \bar{x}' = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix} \bar{x},$$

**Rotation + translation.** This transformation is also known as 2D rigid body motion or the 2D Euclidean transformation (since Euclidean distances are preserved). It can be written as

$$x' = \begin{bmatrix} R & t \end{bmatrix} \bar{x}. \qquad R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

is an orthonormal rotation matrix with RRT = I and |R| = 1.

**Scaled rotation**. Also known as the similarity transform, this transformation can be ex-pressed as x′ = sRx + t, where s is an arbitrary scale factor. It can also be written as

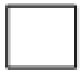
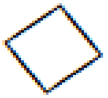
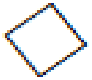
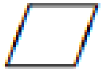
$$\mathbf{x}' = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix} \bar{\mathbf{x}} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \bar{\mathbf{x}},$$

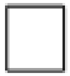
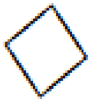
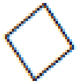
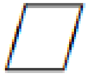
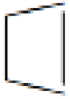
**Affine.** The affine transformation is written as x′ = A¯x, where A is an arbitrary 2 × 3 matrix, i.e.,

$$\mathbf{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \bar{\mathbf{x}}.$$

**Projective.** This transformation, also known as a perspective transform or homography, operates on homogeneous coordinates,

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{H}}\tilde{\mathbf{x}},$$

| Transformation | Matrix | # DoF | Preserves | Icon |
|---|---|---|---|---|
| translation | $\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2\times 3}$ | 2 | orientation |  |
| rigid (Euclidean) | $\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$ | 3 | lengths |  |
| similarity | $\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}$ | 4 | angles |  |
| affine | $\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2\times 3}$ | 6 | parallelism |  |
| projective | $\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3\times 3}$ | 8 | straight lines |  |

| Transformation | Matrix | # DoF | Preserves | Icon |
|---|---|---|---|---|
| translation | $\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{3\times 4}$ | 3 | orientation |  |
| rigid (Euclidean) | $\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3\times 4}$ | 6 | lengths |  |
| similarity | $\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{3\times 4}$ | 7 | angles |  |
| affine | $\begin{bmatrix} \mathbf{A} \end{bmatrix}_{3\times 4}$ | 12 | parallelism |  |
| projective | $\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{4\times 4}$ | 15 | straight lines |  |

## 3D rotations

The biggest difference between 2D and 3D coordinate transformations is that the parameterization of the 3D rotation matrix R is not as straightforward, as several different possibilities exist

### Euler angles

A rotation matrix can be formed as the product of three rotations around three cardinal axes, e.g., x, y, and z, or x, y, and x. This is generally a bad idea, as the result depends on the order in which the transforms are applied.2 What is worse, it is not always possible to move smoothly in the parameter space, i.e., sometimes one or more of the Euler angles change dramatically in response to a small change in rotation

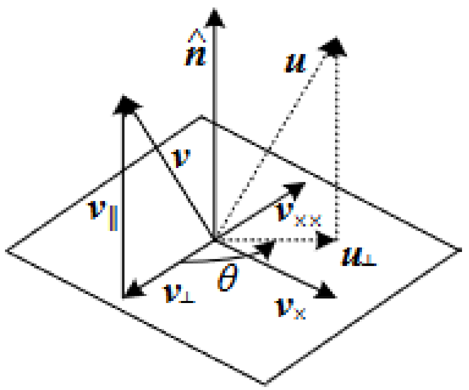
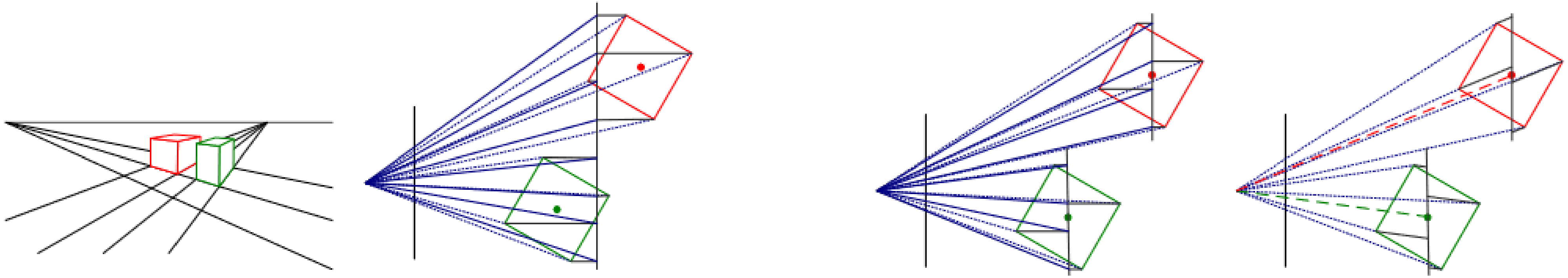


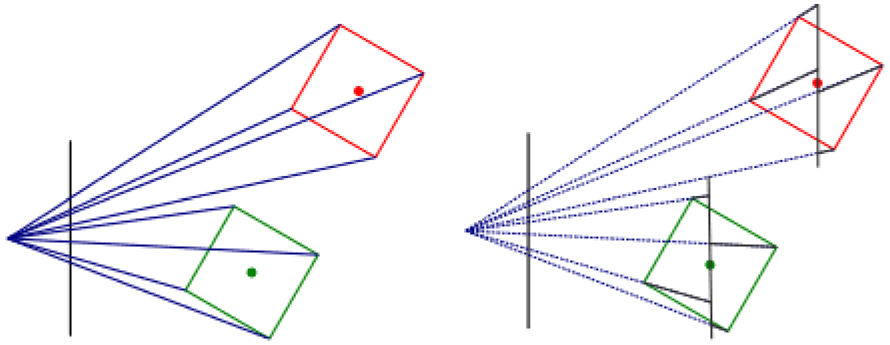

**Figure 2.5** *Rotation around an axis n̂ by an angle θ.*

## 3D to 2D projections

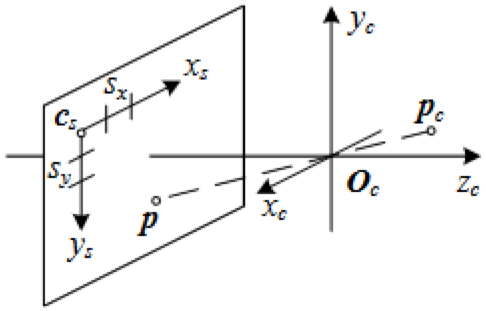
To represent 2D and 3D geometric primitives and how to transform them spatially, we need to specify how 3D primitives are projected onto the image plane. We can do this using a linear 3D to 2D projection matrix. The simplest model is orthography, which requires no division to get the final (inhomogeneous) result. The more commonly used model is perspective, since this more accurately models the behavior of real cameras



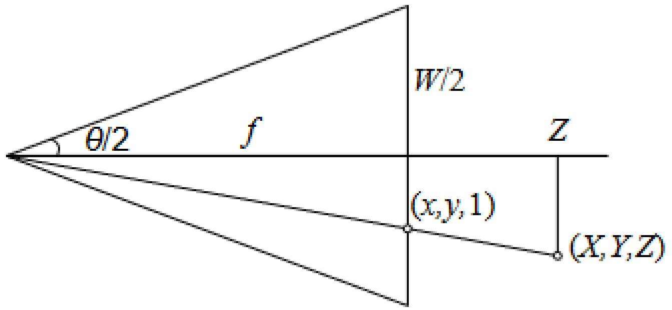
(a) 3D view (b) orthography (c) scaled orthography (d) para-perspective

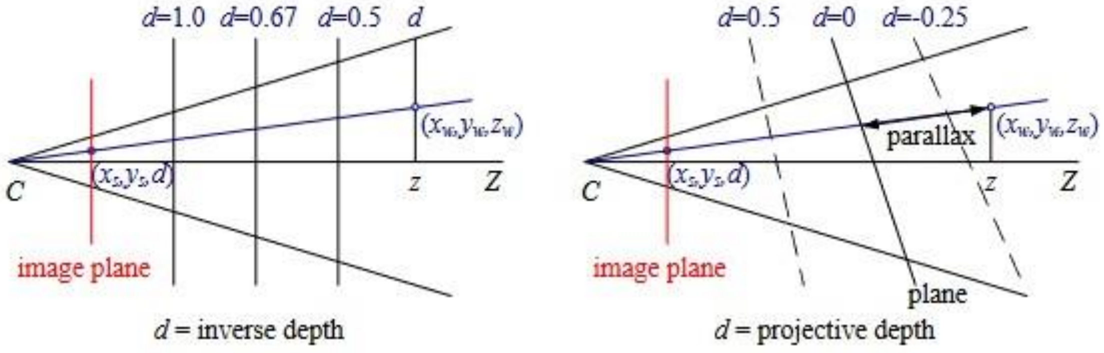


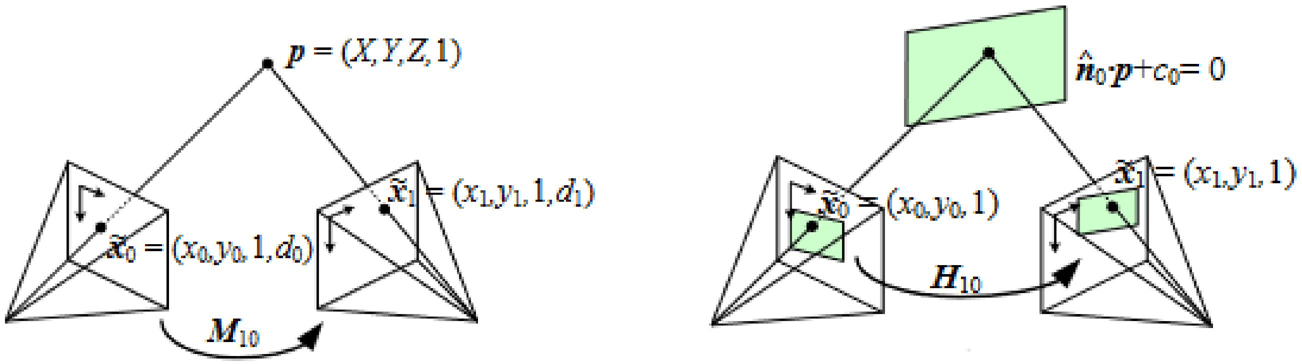
(e) perspective (f) object-centered




Projection of a 3D camera-centered point pc onto the sensor planes at location p. Oc is the optical center (nodal point), cs is the 3D origin of the sensor plane coordinate system and sx and sy are the pixel spacings


W/2
θ/2
f
Z
(x,y,1)
(X,Y,Z)


d=1.0
d=0.67
d=0.5
d
(x_w,y_w,z_w)
C
(x_s,y_s,d)
z
Z
image plane
d = inverse depth
d=0.5
d=0
d=-0.25
parallax
(x_w,y_w,z_w)
C
(x_s,y_s,d)
z
Z
image plane
plane
d = projective depth


p = (X,Y,Z,1)
x̃1 = (x1,y1,1,d1)
x̃0 = (x0,y0,1,d0)
M10
n̂0·p+c0= 0
x̃1 = (x1,y1,1)
x̃0 = (x0,y0,1)
H10

(a) (b)

(a) (b) (c)

**Figure 2.13** *Radial lens distortions: (a) barrel, (b) pincushion, and (c) fisheye. The fisheye image spans almost 180° from side-to-side.*

THANK YOU !!!